PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
H04L 29/06

A1

(11) International Publication Number: WO 98/18248

(43) International Publication Date: 30 April 1998 (30.04.98)

(21) International Application Number: PCT/GB97/02712

(22) International Filing Date: 2 October 1997 (02.10.97)

(30) Priority Data:
08/731,800 21 October 1996 (21.10.96) US

(71) Applicant: INTERNATIONAL BUSINESS MACHINES CORPORATION [US/US]; Armonk, NY 10504 (US).

(71) Applicant *(for MC only)*: IBM UNITED KINGDOM LIMITED [GB/GB]; P.O. Box 41, North Harbour, Portsmouth, Hampshire PO6 3AU (GB).

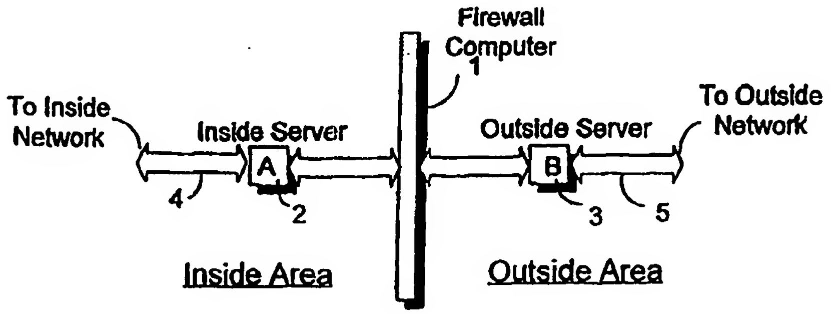
(72) Inventors: JADE, Prashanth; 515 East Henckle Avenue, Ridley Park, PA 19078 (US). MOORE, Victor, Stuart; 4739 Pine Tree Drive, Boynton Beach, FL 33436 (US). RAO, Arun, Mohan; 6301 Shadybrook Lane, Dallas, TX 75206 (US). WALTERS, Glen, Robert; 3208 Astoria Avenue, Sebring, FL 33872 (US).

(74) Agent: MOSS, Robert, Douglas; IBM United Kingdom Limited, Intellectual Property Dept., Hursley Park, Winchester, Hampshire SO21 2JN (GB).

(81) Designated States: BR, CA, CN, CZ, HU, JP, KR, PL, RU, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: OUTSIDE ACCESS TO COMPUTER RESOURCES THROUGH A FIREWALL



(57) Abstract

A firewall isolates computer and network resources inside the firewall from networks, computers and computer applications outside the firewall. Usually, a firewall allows for an inside user or object to originate connection to an outside object or network, but does not allow for connections to be generated in the reverse direction; i.e. from outside in. The disclosed invention provides a special "tunnelling" mechanism, operating on both sides of a firewall, for establishing such "outside in" connections when they are requested by certain "trusted" individuals or objects or applications outside the firewall. The mechanism includes special tunnelling applications, running on interface servers inside and outside the firewall, and a special table of "trusted sockets" created and maintained by the inside tunnelling application.

Summary of the Invention

According to various aspects, the invention provides a tunnelling apparatus, method and computer program product, as claimed in the independent claims, with advantageous preferred features claimed in the dependent claims.

In accordance with the invention, means are provided inside and outside a firewall for cooperatively producing tunnelling effects, in response to certain types of requests initiated by objects outside the firewall, which effects result in creation of connections between such outside objects and resources inside the firewall. Connections so created have the unique property that they are effectively created from "inside out" as if they were requests originating from objects inside the firewall to destinations outside the firewall.

The "types of requests" accommodated by such "tunnelling" means are requests addressed to what are presently termed "trusted sockets". Trusted sockets are entries in a table of trusted sockets that is created and maintained exclusively inside the firewall. Each entry in that table includes the address of a "trusted" port, a protocol (e.g. a telecommunication protocol such as TCP/IP, NNTP, etc.) pre-associated with that address, and the identity of a host object inside the firewall (e.g. a host computer or a host application). Thus, it is understood that in order for an individual and/or object outside the firewall to initiate such a request that individual must be entrusted with the information that represents a trusted socket entry that is currently valid.

The table of trusted sockets is created and maintained by a "tunnelling application" running on an inside interface server (under control of appropriately authorized individuals having direct access to that server) that interfaces between this tunnelling application and all other "accessible" objects/resources inside the firewall (including other applications running on the inside interface server). The inside interface server also establishes a "control connection" to an outside interface server which interfaces between the firewall and all objects outside the firewall. The control connection is accessible only to the tunnelling application running on the inside interface server and a corresponding tunnelling application running on the outside interface server; i.e. it is not directly accessible to any other applications running on these interfaces servers, and is totally inaccessible to both inside and outside objects not residing on these servers.

A copy of the trusted sockets table is transferred from the inside interface server to the outside interface server; e.g. when the table is created and/or altered, or at special times of day, etc.

When an outside object, that is currently not connected through the firewall, originates a request reaching the outside interface server, the tunnelling application on that server determines if the request is directed to a trusted socket entry that is currently valid. If it is not so directed, the request is ignored. If the request is to a trusted socket, the request is passed over the control connection to the tunnelling application on the inside interface server. Concurrently, a process (or task) associated with the request is generated in the outside interface server, and an outside connection is established between that process/task and the requesting object.

Upon receiving the request, the inside tunnelling application also may be required to verify that the request is to a currently valid trusted socket and disallow the request if it is not. If the request is to a currently valid trusted socket, the inside tunnelling application generates (or "spawns") an inside process associated with the request. Then the inside tunnelling application: (a) generates connections between the inside resource associated with the port and host identity of the "requested" trusted socket entry and the inside interface server; and (b) communicating over the control connection with the outside tunnelling application and a computer controlling the firewall itself, generates a connection through the firewall between the tasks generated/spawned on both the inside and outside interface servers. The connections generated/spawned by the inside and outside tunnelling applications are separate from the control connection, and useful to carry data (usually in packet format defined by the trusted socket protocol) bidirectionally between the outside object that originated the request and the inside object targeted by the request.

These and other features, advantages, objectives and benefits of the present invention will be more fully understood by considering the following detailed description and claims.

Brief Description of the Drawings

Figure 1 is a schematic of a typical firewall environment in which the present invention can be applied.

Figure 2 is a flow diagram illustrating the creation and handling of the trusted socket table mentioned above.

Figure 3 is a flow diagram illustrating the firewall tunnelling process of the present invention.

Figure 4 illustrates a preferred form of the trusted sockets table mentioned above.

Figure 5 is a flow diagram for explaining details of tunnelling application operations inside and outside of a firewall, in accordance with the present invention.

Detailed Description of the Preferred Embodiments

Figure 1 illustrates a typical firewall environment for application of the present invention. Firewall computer 1 maintains the firewall security function in accordance with presently commonplace procedures. The functions of this computer, other than those involving extending connections from objects inside the firewall to objects outside the firewall, are transparent to (and in essence not relevant to) the present invention. Interface servers 2 and 3 (labelled servers A and B respectively) operate respectively inside and outside the firewall created by 1. Server A interfaces between the firewall and objects (software applications, hardware entities, etc.) inside the firewall, including objects in Server A itself. Server B interfaces between the firewall and objects outside the firewall, including objects in server B itself.

In a typical firewall usage environment, server A connects to a network inside the firewall (e.g. a private local area network) via a connection shown at 4, and server B connects to a network outside the firewall (e.g. the Internet) via a connection shown at 5.

In applying the present invention to this environmental configuration, servers A and B are provided with "tunnelling" software applications and store copies of a "trusted socket" table. These entities -- the tunnelling applications and the trusted socket table -- are considered unique to the present invention and described herein.

Figures 2 and 3 describe (tunnelling) processes performed at servers A and B in furtherance of the present invention.

As shown at 10, in Figure 2, a trusted socket table (which is described below in reference to figure 4) is created in and stored at server A (or a store readily accessible to that server). As shown at 11, server A creates a special "control connection" to server B through the firewall (computer), and passes a copy of the trusted sockets table to server B over the control connection. This control connection, also considered part of the present invention, is used by the above-mentioned tunnelling applications to effectively inter-communicate, and thereby form other connections (hereinafter termed "data connections") between objects inside and outside the firewall, in response to requests received from outside objects.

Segments of these data connections extending through the firewall are entirely separate from the control connection used in their formation, and are always formed under control of processes running inside the firewall. For an outside request to give rise to formation of a data connection to an inside object, the request must be directed to an entry in the trusted sockets table, and validated as such. Outside requests found to be invalid are ignored, so that the firewall and its inside resources are effectively invisible to and inaccessible to outside requesters having invalid request information. Conversely, it should be understood that valid requests are issuable only at the direction of individuals having privileged knowledge of currently valid entries in the trusted sockets table (e.g. telecommuting employees of the owner of the inside resources, etc.).

Figure 3 describes tunnelling functions performed at servers A and B, after B has received and stored its copy of the trusted sockets table sent by A.

As shown at 20, (the tunnelling application in) server B waits to receive an outside request that effectively calls for a tunnelling operation; i.e. creation of a data connection between an inside "host" object designated in the request and the outside object from which the request was sent. Upon receiving a request (21, fig. 3), (the tunnelling application at) B checks to verify that the request is a valid one (decision 22, fig. 3). In respect to the last-mentioned function, it should be understood that server B only receives requests directed to that server, and that the tunnelling application on server B only receives requests that appear to be directed to a port inside the firewall, and distinguishes those requests as valid only if they are directed to a currently valid entry in the trusted sockets table mentioned earlier.

If the request is invalid it is ignored, and (the application at) server B resumes waiting for a request. However, if the request is valid, (the tunnelling application at) server B creates a process or task "B.1" for handling outside elements of data transfer relative to the requesting object (23, fig. 3). Task B.1 establishes a data connection between itself and the requesting object (also 23, fig. 3), and forwards the request to (the tunnelling application at) server A, via the control connection, along with the identity of task B.1 (24, fig. 3).

Upon receiving a validated request, (the tunnelling application at) server A generates a process or task A.1, for handling inside aspects of the transmission of data between the outside requesting object and a host object identified in the request (25, fig. 3; the latter object being a component of a trusted socket designation as explained below). Task A.1 creates data connection segments from the host object to the firewall computer (also 25, fig. 3), and instructs the firewall computer to form a connection to B.1 (also 25, fig. 1); thus completing a data connection between the inside host object and the outside requesting object. It should be appreciated that this data connection may require buffers, in servers A and B and the firewall computer, of a size determined by the protocol of data transmission (discussed further below), and the required speed of (packet) transfer for that protocol.

The form of the trusted sockets table is illustrated in Figure 4. Examples of 2 specific entries are shown at 30, and additional entries are implied at 31 by dotted lines extending downward from the second entry. Each entry consists of a port number, information defining a transmission protocol (usually, a burst packet transfer protocol), and information identifying a host object. The port number is an address inside the firewall assigned to the host object. As examples of protocols, the first two entries in the table list NNTP (Network News Transport Protocol) and HTTP (HyperText Transport Protocol).

Figure 5 shows in finer detail operations performed by the tunnelling applications at interface servers A and B. Operations that are the same as operations shown in Figures 2 and 3 are identified by identical numerals. Operations that are parts of, or differ in some respect from, operations shown in Figures 2 and 3 are identified by the same numbers followed by letters (a, b, etc.). Other operations are identified by numbers different from those previously used.

Operation 10a at server A, a composite of operations 10 and 12 of fig. 2, is the creation and updating (expansion, modification. etc.) of the trusted sockets table and the copying of the latter to server B. Operation 11a at server A is the establishment or (as explained below) re-establishment of the control connection between (the tunnelling applications at) servers A and B. A need to re-establish the control connection arises when the connection is unintentionally broken, and the operations required to detect and respond to such occurrences are shown at 46-48 in Fig. 5 (which are discussed further below).

After receiving its copy of the trusted sockets table, (the tunnelling application at) server B listens for outside requests (20, fig. 5). When a valid outside tunnelling request is received, and an associated data handling task (e.g. B.1, fig. 3) has been created therefor (21-22a, 24a, fig. 5), server B presents the request to server A (23a, fig. 5), along with control signals indicating the action occurring and information identifying the task (e.g. B.1) created at B to attend to the request. Server B then waits for acknowledgement of receipt of the request from server A (23c, fig. 5), and upon receiving such server B establishes a data connection segment from the newly created task to the requesting object (24b, fig. 5; e.g. from B.1 to C as in fig. 3). Server B then waits for establishment of a data connection segment from the firewall to the task just created at B (24c, fig. 5), that occurrence implying establishment of an associated data connection segment between the host object (the one identified in the request) and server B. The tunnelling process at server B is then complete until the data connection segment between the firewall and the task at B is terminated (40, fig. 5), ending the involvement of server B in that connection and the associated request (41, fig. 5).

Returning to consideration of tunnelling actions at server A, after establishing or re-establishing the control connection, server A listens for (request forwarding) signals from B (46, fig. 5). If a signal hasn't been received (47, fig. 5), but a predetermined timeout interval has not elapsed since the waiting started (48, fig. 5), server A merely continues to wait for such signal. However, if the timeout has lapsed (Yes decision at 48, fig. 5) it is assumed that the control connection has been (unintentionally) broken, and the connection is re-established (11a repeated).

If a request is received from server B, server A may optionally perform its own validation operation (49, fig. 5) to verify that the request is to a currently valid trusted socket. If that option is used and the request is found to be invalid, an error signal would be returned to server B instead of the acknowledgement awaited at 23b. If the option is not used, or if it is used and the request is found to be valid, server A proceeds to establish its internal task such as A.1, and the latter, as described previously, forms data connection segments from the host object to the firewall, and directs the firewall computer to extend the data connection to B.1 (50, fig. 5). This concludes server A's involvement in the current request, freeing it to continue with other requests (51, fig. 5).

Program Products

The foregoing tunnelling applications can be delivered as a "computer readable" program product; e.g. on storage media or through communication networks. It should be understood that such product can be

Fig. 1

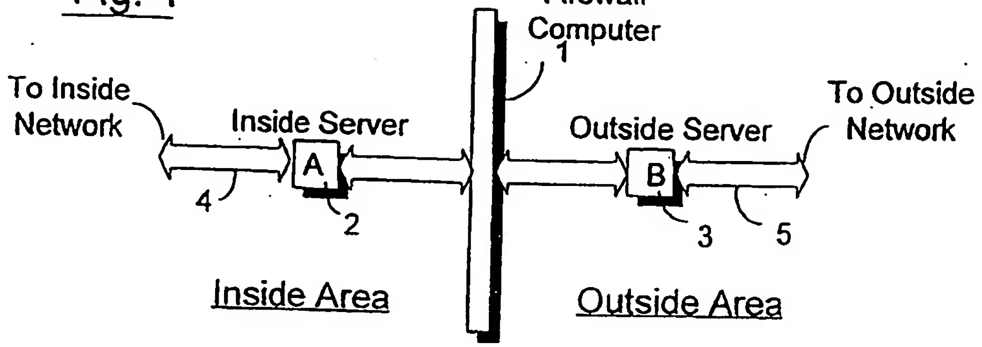


Fig. 2

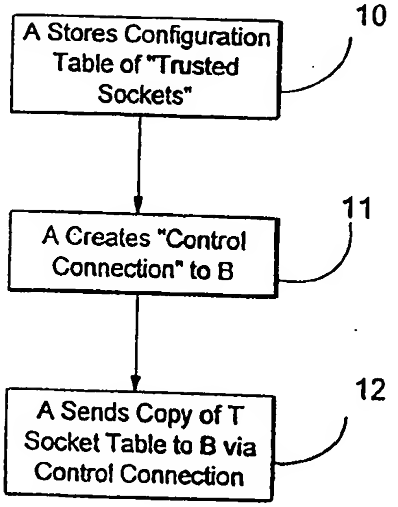

A Stores Configuration Table of "Trusted Sockets"
10
A Creates "Control Connection" to B
11
A Sends Copy of T Socket Table to B via Control Connection
12

Fig. 3

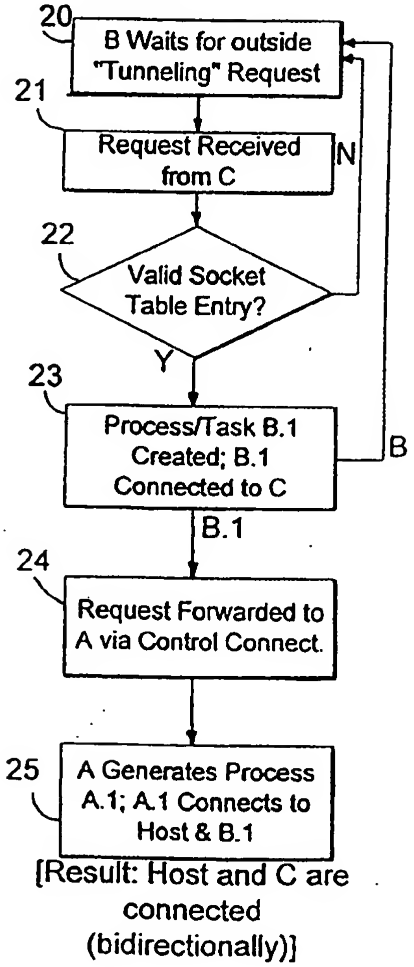
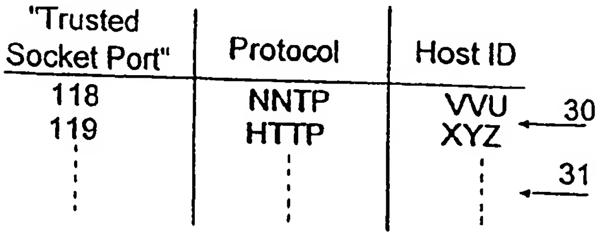

20
B Waits for outside "Tunneling" Request
21
Request Received from C
N
22
Valid Socket Table Entry?
Y
23
Process/Task B.1 Created; B.1 Connected to C
B
B.1
24
Request Forwarded to A via Control Connect.
25
A Generates Process A.1; A.1 Connects to Host & B.1
[Result: Host and C are connected (bidirectionally)]

Fig. 4



"Trusted Socket Port"	Protocol	Host ID
118	NNTP	VVU
119	HTTP	XYZ
⋮	⋮	⋮